Flexible Protein Folding by Ant Colony Optimization

Xiao-min Hu[1], Jun Zhang[1] (Corresponding Author), and Yun Li[2]

1 Department of Computer Science, SUN Yat-sen University, Guangzhou, 510275, China. junzhang@ieee.org

2 Department of Electronics and Electrical Engineering, University of Glasgow, Glasgow G12 8LT, Scotland, UK, Yun.Li@elec.gla.ac.uk

(This work was supported by the Natioinal Science Foundation (NSF) of China (60573066) and the NSF of Guangdong (5003346), China and SRF for ROCS, SEM, China)

1 Introduction

With rapid development of bioinformatics, more and more about the molecular world becomes known. Following the completion of the human genome project in 2000, genetic sequencing is now made feasible by current technology [1]. However, there still exist challenges in analyzing relationships between protein structures and their related functions. As different structures reflect specifically different functions, predicting a protein structure to estimate its functions is one of the major goals of bioinformatics [2]. In nature, proteins fold spontaneously to their native structures very fast (on a time scale of milliseconds) when placed in an aqueous solution [3]. However, traditional methods for predicting the structures of proteins, such as the X-ray crystallography and the nuclear magnetic resonance (NMR) [4][5] are expensive and time-consuming. More importantly, reflections that are gained by these methods may be blurry and incomplete. Since the remarkable discovery by Anfinsen et al [6] that many simple proteins have a unique native structure, which appear to depend on the sequence only, experimental results [7][8][9] have subsequently emerged to support this discovery. A commonly accepted hypothesis is that the protein sequence folds into the structure with the equilibrium minimum free energy (MFE) state (the thermodynamic hypothesis) [10][11]. Given a two-dimensional square lattice board, the protein folding problem (PFP) is to place the protein sequence in the lattice to form a self-avoiding path. Thus, the aim of solving a PFP is to find the protein folding conformation that satisfies the MFE state.

Based on this hypothesis, the critical mission is to find a way to predict a protein structure fast and accurately from the protein sequence. The real structures of proteins are very complex for they are on the atomic level and in a relatively large (?) three dimensional search space. Various protein folding models have been proposed to simplify the structure for better analysis. These models include the protein structure prediction (PSP) model [12], the lattice polymer embedding (LPE) model [10], the charged graph embedding (CGE) model [13], and the hydrophobic-hydrophilic (or hydrophobic-polar, HP) model [14]-[23], etc. In particular, the HP model can be further classified into three types - the square lattice model [14]-[18], the triangle lattice model [19]-[21], and the toy model [22][23]. The algorithm proposed in this chapter is based on the HP square lattice model.

Since a number of simplified models have been proposed, various methods have been developed to solve the PFPs, such as the dynamic programming (DP), neural network (NN) [23], Monte Carlo (MC) [24]-[28], genetic algorithm (GA) [29]-[36], ant colony optimization (ACO) [37]-[41], particle swarm optimization (PSO) [22], and immune algorithm (IA) [42][43] methods. The PFPs are proven to be NP-complete [44], which cannot be solved by a deterministic polynomial algorithm. As a paradigm of swarm computation, ant colony algorithms [45] have shown great potential in solving NP-hard

combinatorial problems.

This chapter develops a simple but effective ant algorithm to solve PFPs, termed the 'flexible ant colony (FAC) algorithm'. It has four special mechanisms, including the path construction, the path retrieval, the pheromone attraction, and the folding heuristics. These novel mechanisms make it behave differently from previous ant algorithms for solving PFPs with the HP square lattice model [37]-[41].

The ants of the FAC algorithm aim to find a 'conformation path' of protein in the lattice. The pheromones are deposited on the virtual connections between adjacent squares in the lattice. Such pheromone laying approach is similar to those on the arcs connecting cities in a traveling salesman problem (TSP). However, it is different from existing ant algorithms proposed for solving PFPs, whose pheromones are on three relative folding directions of the protein [37]-[41]. In fact, if the pheromones only indicate the relative directions of the protein folding (as the ones in [37]-[41] do), the ants may not grasp the folding situations entirely. On the contrary, if the pheromones guide the protein to fold on an absolute lattice, as the proposed algorithm does, each ant can sense the solutions which have been configured by the other ants.

A protein sequence in the HP square lattice model is a string of hydrophobic (H) and polar (P) amino acids. The amino acids are placed one by one by artificial ants. If the surrounding lattice squares of an amino acid are all occupied, the next amino acid cannot be placed. Such situation is termed stagnation. Then the path retrieving strategy should be applied. As all ants start to construct the folding path from the center of the lattice, diversity for ants to choose alternative squares to place the amino acid on the protein sequence is realized by decreasing the pheromone value on the arc that the ant has just passed. Such pheromone reduction method during solution construction is similar to the local search method used in the ant colony system (ACS) algorithm [45][46].

In the proposed FAC algorithm, the heuristics and the pheromones cooperate to construct conformations. The heuristic information varies between the hydrophobic amino acids and the polar amino acids. An added local search method is optional and it is omitted in this chapter, as the performance of the FAC without local search is shown good enough in most of the experimental tests. By comparing the performance with a genetic algorithm (GA), an immune algorithm (IA), and an ant algorithm in the literature, the proposed algorithm does present improvements.

The rest of this chapter is constructed as follows: Section 2 presents a brief review on the PFPs with the HP model and discusses the features of protein folding conformations. Then the characteristics of the ACO are briefly introduced. Section 3 details the ants' construction behaviors of the proposed FAC algorithm. Section 4 folds some benchmark protein sequences using the proposed algorithm and compares the performances with other well-known algorithms. For deeper analysis, this chapter also tests influences of the parameters of the FAC algorithm and highlights some prospects for enhancements. Finally, conclusions are drawn in Section 5.

2 Protein Folding and the Ant Colony Optimization

2.1 Characteristics of the Protein Folding

Some benchmark instances of protein sequences in the 2-dimensional square lattice HP (2D-HP) models are listed in Table 1, where l is the number of amino acids and E^* stands for the MFE level. The letter 'H' stands for the hydrophobic amino acid and 'P' stands for the polar amino acid, which is hydrophilic. There are 20 amino acids in nature. Using various classifications, they can be divided into acid, alkaline or neutral; positively or negatively charged or uncharged; and hydrophobic or hydrophilic, etc. In a globular protein in an aqueous solution, the hydrophilic amino acids tend to be on

the surface of the globule as they are attracted to water molecules (note that the environment inside cells is primarily water). The hydrophobic amino acids are repelled by water so that most of them gather inside the globular protein to form a core except for some special hydrophobic regions on the surface of the protein.

Table 1. Standard HP benchmarks for 2-D square lattice

No.	*l*	E^*	Protein Sequence
1	18	-9	PHPPHPHHHPHHPHHHHH
2	18	-8	HPHPHHHPPPHHHHPPHH
3	20	-10	HHHPPHPHPHPPHPHPHPPH
4	20	-9	HPHPPHHPHPPHPHHPPHPH
5	24	-9	HHPPHPPHPPHPPHPPHPPHPPHH
6	25	-8	PPHPPHHPPPPHHPPPPHHPPPPHH
7	36	-14	PPPHHPPHHPPPPPHHHHHHHPPH HPPPPHHPPHPP
8	48	-23	PPHPPHHPPHHPPPPPHHHHHHHH HHPPPPPPHHPPHHPPHPPHHHHH

Some conformations of the 2D-HP folding structures of the same protein sequence with 48 amino acids are presented in Fig. 1. For the square lattice HP model, the best conformation is judged by the number of hydrophobic-hydrophobic (H-H) bonds that hydrophobic amino acids are adjacent on the lattice, but not consecutive in the sequence. The number of the H-H bonds in each conformation in Fig. 1 is 23 (e.g., the number of dashed lines in the first conformation), which forms the MFE state with $E^* = -23$. It can be seen that the hydrophobic amino acids do form a core inside the protein conformation, while the polar amino acids are surrounding the core and their placements are quite flexible.

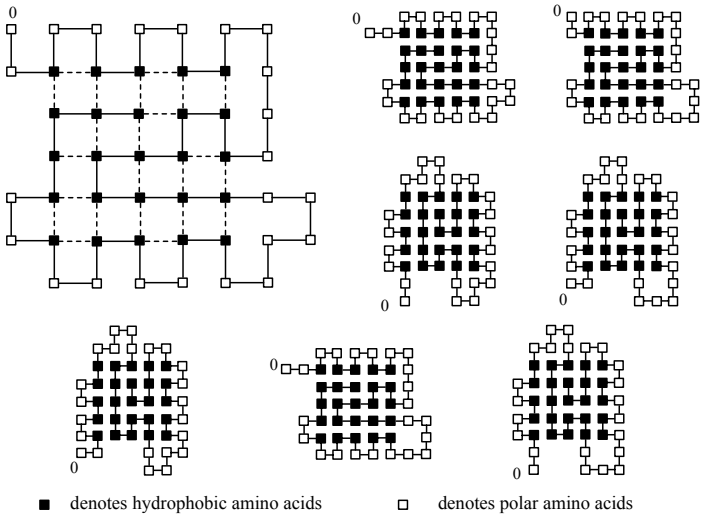


Fig. 1. Some conformations of sequence 8 (Length = 48)

Although the square lattice model is highly abstracted from the real protein folding model, some special conformations can reflect a possible secondary structure of a protein. Fig. 2 shows a special 2D-HP conformations and the corresponding three-dimensional protein structures of an α-helix and

β -sheets. However, such a model is unsatisfactory to many biologists. The 3-dimensional structure of a specific protein sequence is unique, but as we can see in Fig. 1, there may be several equivalent conformations by the same protein sequence. Therefore, the HP model is too simple to reflect the real protein structure completely. However, it is already a very challenging computational model.

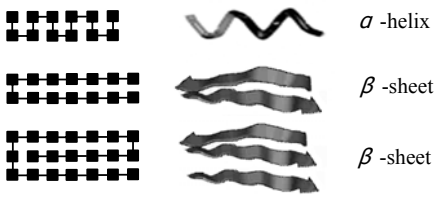


Fig. 2. Special HP conformations and the secondary structures of protein sequences

2.2 Characteristics of the Ant Colony Optimization

To solve the traveling salesman problem, the first ant algorithm - the ant system (AS) - was proposed by Dorigo [45][47] through stimulating the foraging behavior of real ants. Following this, several variants of ant algorithms have been developed, such as the elitist ant system (EAS), the max-min ant system (MMAS), and the ant colony system (ACS) [45], etc. They have been successfully applied to a wide range of application problems, such as the vehicle routing problem (VRP) [48], the job shop scheduling problem (JSP) [49], and the water distribution system (WDS) [50], etc. The AS and its successors at last form a kind of optimization paradigm termed the 'ant colony optimization (ACO) algorithms'. The basic framework for ACO includes:

Step 1: Construct ants' solutions (utilizing pheromone and heuristic information)

Step 2: Apply local search (optional)

Step 3: Update pheromones

A group of m ants perform the above three steps to search for a better solution iteration by iteration. Firstly, based on the current density of pheromone in the environment and other heuristic information, each ant in the colony constructs a solution. Secondly, local search method can be applied to enhance the solutions found by the ants. Thirdly, the pheromone in the surrounding environment should be updated to guide more ants to the potentially best solution in the next iteration.

The FAC algorithm proposed in this chapter is based on the basic framework of the ant colony system (ACS) [45][46], which includes such mechanisms as local pheromone update and global pheromone update. The implementation of these mechanisms is redefined in this chapter.

3 Ant Colony Search in Lattices

Given a two-dimensional square lattice board, the PFP is to place the protein sequence in the lattice to form a self-avoiding path. The mission of an ant colony is to discover a path, which maximizes the number of H-H bonds by two adjacent hydrophobic amino acids that are not consecutive in the protein sequence.

3.1 Path Construction

In order not to violate the region of the lattice, each ant starts building the path from the middle of the protein sequence in the center of the lattice. For a protein sequence with n amino acids, which are denoted as $\{s_0, s_1, \ldots, s_{n-1}\}$ ($s_j \in \{P, H\}, j = 0,...,n-1$), each ant starts from two horizontal squares in

the middle of an (n+2)×(n+2) lattice board, as depicted in Fig. 3. The squares in the lattice board are indexed from 0 to (n+2)^2-1, starting from the top left corner. The two squares with indexes $\left(\lfloor n/2 \rfloor+1\right)\cdot\left(n+2\right)+\lfloor n/2 \rfloor$ and $\left(\lfloor n/2 \rfloor+1\right)\cdot\left(n+2\right)+\lfloor n/2 \rfloor+1$ are termed the 'left start square' and the 'right start square', respectively. The two squares are colored in the middle of the lattice shown in Fig. 3. The amino acid $s_{\lfloor n/2 \rfloor}$ is placed in the left start square while the amino acid $s_{\lfloor n/2 \rfloor+1}$ is placed in the right start square. The sub-protein sequence $\{s_0,...,s_{\lfloor n/2 \rfloor}\}$ that is built from the left start square is denoted as the 'left path', while the $\{s_{\lfloor n/2 \rfloor+1},...,s_{n-1}\}$ is the 'right path'. Then an ant randomly chooses to go a step on the left part or on the right part of the protein sequence. After several construction steps, a protein sequence is built, similar to the dashed lines in Fig. 3. The squares that have been passed by the ant cannot be passed again by the same ant.

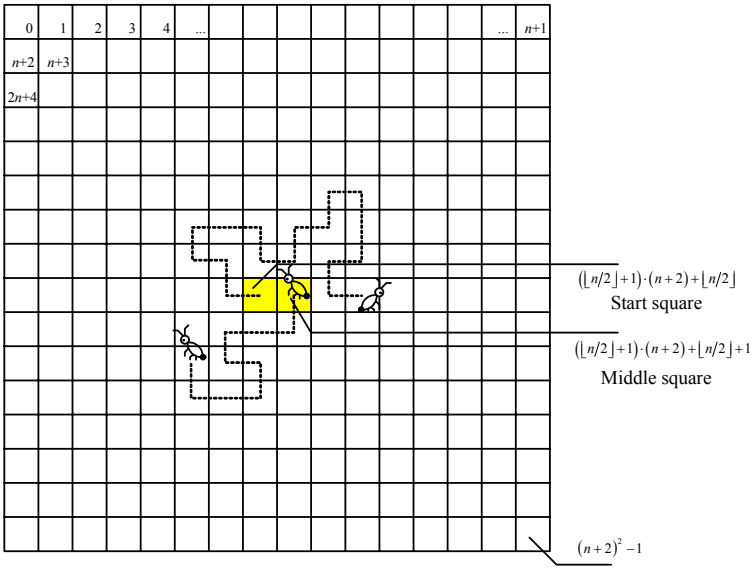


Fig. 3. Lattice board for a protein with n amino acids

There are two advantages of indexing the squares in the lattice. One is that the coordinates of the squares are now one-dimensional. The other is that the four adjacent squares are convenient to obtain. For example, when an ant is now in square i as shown in Fig. 4, which is not on the border of the lattice, its four adjacent squares are i-(n+2) (going up (U)), i+(n+2) (down (D)), i-1 (left (L)), and i+1 (right (R)). As the ant has passed the right square, it can only choose one of the other three directions to go.

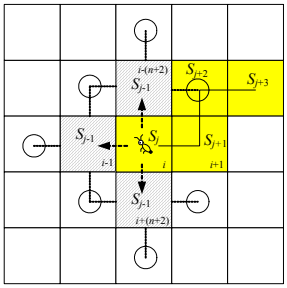


Fig. 4. An ant chooses a step to go

3.2 Path Retrieval

If the ant has passed all the adjacent squares when placing a non-ending amino acid $s_j (j \neq 0 \text{ or } n-1)$, the protein cannot fold any more. Such situation is termed 'stagnation'. In this case, the folding needs to be retrieved. Consider an ant has constructed a sub-sequence $\{s_{left}, \ldots, s_{startL}, s_{startR}, \ldots, s_{right}\}$, where *left* is the index of the left most amino acid, *right* is the index of the right most amino acid, *startL* = $\lfloor n/2 \rfloor$, *startR* = $\lfloor n/2 \rfloor + 1$. To a 'right' retrieval, a random index *j* is selected as

$$j = rand \ \% \ (right - start - 1) + start + 1 \quad (1)$$

where *rand* is a random non-negative integer number. The amino acids from s_{j+1} to s_{right} are released as not been constructed by the ant and the corresponding squares in the lattice are set vacant. On the other hand, a 'left retrieval' point *j* is selected as

$$j = rand \ \% \ (start - left) + left + 1 \quad (2)$$

The amino acids from s_{left} to s_{j-1} are released and the corresponding squares are thus set vacant.

Although stagnation occurs on the right side of the protein, it doesn't mean that the right side of the protein is to be retrieved, because certain stagnation cannot be cleared by simply retrieving the side where the stagnation happens. Fig. 5 illustrates two stagnation situations of the right path. The hollow beads stand for the left start amino acid and the right start amino acid, while the triangles are amino acids on the right path. In the example presented in Fig. 5(a), the stagnation can be released by the right retrieval when its point is *j*= 21 and the next direction to fold is upward.

However, in Fig. 5(b), the stagnation cannot be released by performing right retrieval but only left retrieval. So the Boolean values *RightRetrieveBool* and *LeftRetrieveBool* are used to judge such situations to make sure that a retrieval in the same direction cannot be performed twice consecutively for avoiding potential stagnation.

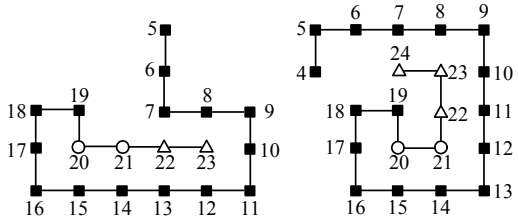


Fig. 5. Examples of the stagnation

Whether to perform a right retrieval or a left retrieval is not only based on the location of the stagnation, but also the two Boolean values *RightRetrieveBool* and *LeftRetrieveBool*. If the stagnation happens on the right side of the protein, we term the retrieval procedure as 'RightSideRetrieve', while the procedure for the left stagnation is termed 'LeftSideRetrieve'. Fig. 6 illustrates the pseudo-code of the above process. The functions 'RightRetrieveSequence( )' and 'LeftRetrieveSequence( )' perform the respective right/left retrieval. Take Fig. 5(b) as an example. The stagnation happens on the right path, so that the 'RightSideRetrieve' procedure is invoked. As *startR* = 21, *right* = 24, and RightSideRetrieve =false, a random integer *j* is generated by (1). Suppose *j*= 22. Then the 'RightRetrieveSequence' function is invoked, so that the amino acids from 23 to 24 are released. The 'LeftRetrieveBool' and the 'RightRetrieveBool' are set as False and True, respectively. It is known that this could not help to clear the stagnation. The construction of the path continues, until the stagnation happens again. Suppose the sequence is changed to be 2 to 24. At that time, the 'RightSideRetrieve'

procedure is invoked again. As 'RightRetrieveBool' is true now, it can only perform 'LeftRetrieveSequence( )' to release some of the left path. The stagnation can be cleared if j= 5 to 20.

```
/* startL = ⌊n/2⌋, startR = ⌊n/2⌋+1
   left : the constructed left end, right: the constructed right end */
Procedure RightSideRetrieve({s_left, ⋯ , s_startL, s_startR, ⋯ , s_right})
   If startR < right && RightRetrieveBool == false
         j = rand % (right – startL – 1) + startL +1;
         RightRetrieveSequence({s_left, ⋯ , s_startL, s_startR, ⋯ , s_right}, j);
         LeftRetrieveBool = false;
         RightRetrieveBool = true;
   Else If startL != left
         j = rand % (startL – left) + left +1;
         LeftRetrieveSequence({s_left, ⋯ , s_startL, s_startR, ⋯ , s_right}, j);
         RightRetrieveBool = false;
   End

Procedure LeftSideRetrieve({s_left, ⋯ , s_startL, s_startR, ⋯ , s_right})
   If startL > left && LeftRetrieveBool == false
         j = rand % (startL – left) + left +1;
         LeftRetrieveSequence({s_left, ⋯ , s_startL, s_startR, ⋯ , s_right}, j);
         LeftRetrieveBool = true;
         RightRetrieveBool = false;
   Else If (startL+1) < right
         j = rand % (right – startL – 1) + startL +1;
         RightRetrieveSequence({s_left, ⋯ , s_startL, s_startR, ⋯ , s_right}, j);
         LeftRetrieveBool = false;
   End
```

Fig. 6. Outline of retrieval process

3.3 Pheromone Attraction

Pheromones are released on the directed arcs connecting the adjacent squares, which are denoted as τ_{id}, where $i = 0,1,2,...,(n+2)^2 - 1$ and $d = \{L, R, U, D\}$. Note that the protein sequence cannot exceed the lattice board, as the width of the board must be greater than the length of the protein.

1) Local Pheromone Update

The squares in the lattice board can be compared to cities in the TSP, in which the pheromones are deposited on the arcs connecting the cities. As all ants start from the same left-start and right-start squares in the lattice, an effective method for avoiding early convergence is to remove some pheromones on the arcs that an ant just chose, i.e.,

$\tau_{id} = \delta \cdot \tau_{id}$ (suggest to replace = with <- or replace the left tau with tau', for this is suppo sed to be a math formula, and not computer code) (3)

where $i = 0,1,2,...,(n+2)^2 - 1$, $d = \{L, R, U, D\}$, $\delta = (m-1)/m < 1$ is a 'local evaporation rate', and m is the number of ants. If the pheromone on that arc is smaller than $\tau_{\min}$, the pheromone is reset to $\tau_{\min}$, which is the lower bound of the pheromone value.

2) Global Pheromone Update

Once all ants have constructed a protein folding path, the pheromones on all arcs 'evaporate' as defined by

$$\tau_{id} = \rho \cdot \tau_{id} \quad \text{(see note on eq.3)} \tag{4}$$

where ρ is a 'global evaporation rate'. Then the best path found in the current iteration is reinforced by increasing the amount of pheromone

$$\tau_{i'd} = \tau_{i'd} + \varepsilon / (-E^*_{\min}) \quad \text{(see note on eq.3)} \quad (5)$$

where $i' \in$ {the squares that the iteration's best ant has just passed}, $d = \{L, R, U, D\}$, ε. is the maximum number of H-H bonds in the current iteration, $E^*_{\min} < 0$ is the approximation of the MFE of the protein in the square lattice HP model.

3.4 Heuristics for Folding

While pheromones are the means for keeping the historical memories, heuristics are the strategies for current selection. Different from the heuristic information in [37]-[41] where only hydrophobic amino acids are considered, this chapter takes into account both the heuristic information for hydrophobic amino acids and polar amino acids.

1) Heuristic for hydrophobic (H) amino acids

The goal for PFPs is to find the minimum energy conformation, which is reflected by the number of H-H bonds. Hence, if a conformation can yield more H-H bonds, it should have a higher probability to be constructed. Once the next amino acid s_j. for ant k to place is known as a hydrophobic (H) amino acid, the heuristic for it is determined by

$$\eta_{j-1,l} = h_{j-1,l} + 1 \quad (6)$$

where h_{jl}. is the number of H-H bonds for a possible location of the next amino acid to obtain (excluding consecutive hydrophobic amino acid), $l \in \{1, ..., fea(s_j)\}$ and $fea(s_j)$ is the number of feasible locations for s_j. ($1 \le fea(s_j) \le 3$) (if l is not used in eq.6, this yellow stripped part should be deleted). Fig. 4 illustrates an ant that is currently locating in square I with an amino acid s_j. The next step it chooses is to place an amino acid s_{j-1}. The dashed squares are the possible locations. For each of the slashed squares, the potential H-H bonds are the ones that connect the neighboring squares (shown as hollow spots in Fig. 4) where a hydrophobic amino acid has been placed.

2) Heuristic for polar (P) amino acids

If the next amino acid s_j to be placed is a polar amino acid, the heuristic value is the sum of the vacant squares (i.e., the squares that have not been passed by the ant) and polar amino acids (excluding consecutive polar amino acid) in the neighborhood of the possible location of the next amino acid plus one as given by

$$p = \frac{\tau_{jl} \cdot \eta_{jl}^{\beta}}{\sum_{q \in \{1, ..., fea(s_j)\}} (\tau_{jq} \cdot \eta_{jq}^{\beta})} \quad (7)$$

where v_{jl}. and h'_{jl}. are the numbers of vacant squares and polar amino acids in the neighborhood of the possible locations of the next amino acid, respectively.

For a polar amino acid, more inclinations should be given to water molecules. As the protein folds in an aqueous solution, the vacant squares can be regarded as water molecules. The nearby polar amino acids imply that the edge of the protein is near.

3.5 Implementation of the Flexible Ant Colony Algorithm for PFPs

The roulette wheel selection method is used for each ant in the colony to choose the next step of path. If the current amino acid is s_j. and the next amino acid to be placed is s_{j-1} and the number of feasible directions to fold is $fea(s_{j-1})$, then the probability of selecting the *l*th direction is given by

$$p_l = \frac{\tau_{jl} \cdot \eta_{jl}^{\beta}}{\sum_{q \in \{1,\ldots,fea(s_j)\}} (\tau_{jq} \cdot \eta_{jq}^{\beta})} \tag{8}$$

where $l \in \{1,\ldots,fea(s_j)\}$ and β is the reinforcement to heuristic values.

Generate a random value r ($r \in (0,1]$). If an integer k satisfies (9), the *k*th feasible direction is selected.

$$\sum_{i=1}^{k-1} p_i < r \le \sum_{i=1}^{k} p_i \tag{9}$$

Note that $\sum_{i=1}^{k-1} p_i = 0$ when $k = 1$.

The implementation of the FAC algorithm can be realized as follows:

Step 1: Read in the protein sequence and initialize the parameters.

Step 2: Place all ants in the left start square and the right start square in the lattice.

Step 3: All ants construct feasible folding conformations to the input protein sequence.

Step 4: Evaluate the constructed folding paths and select the best ant in an iteration.

Step 5: Perform a global pheromone update.

Step 6: If the terminate condition is not met, go to Step 2; else terminate the algorithm.

A more detailed flowchart of the proposed algorithm is illustrated in Fig. 7.

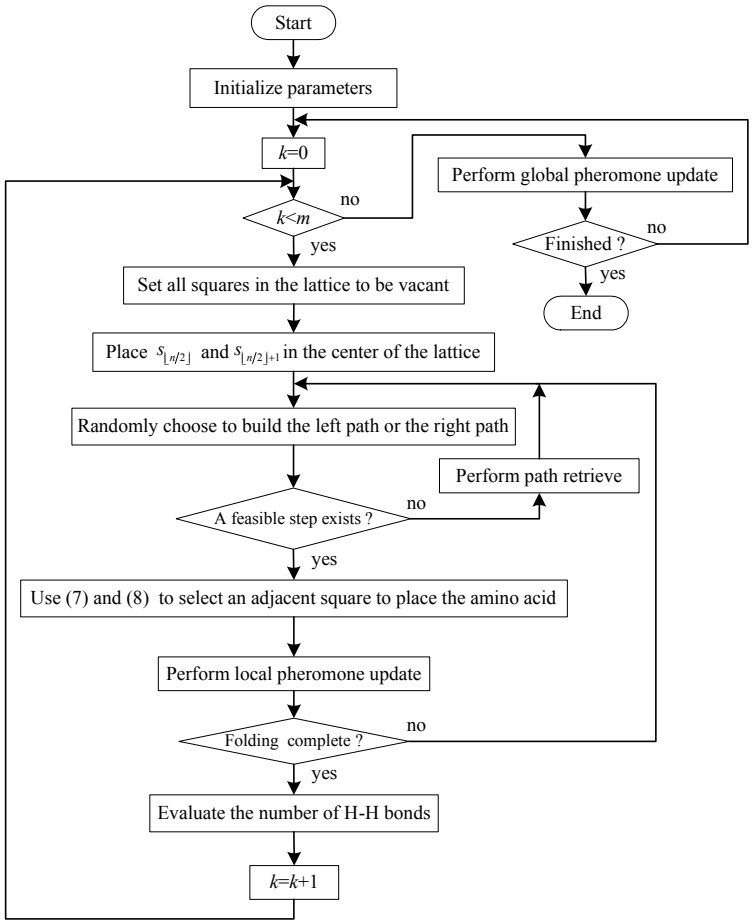


Fig. 7. Flowchart of the FAC algorithm

4 Experiments and Discussions

The benchmark instances of HP protein folding are tabulated in Table 1. The parameters' settings for the proposed FAC algorithm are $\tau_0 = 1/3$, $\tau_{\min} = 0.05$ and $\rho = 0.9$. For sequences 1–7, m = 10 and $\beta = 2$. For sequence 8, m = 100 and $\beta = 3$. Each group of parameters has been tested 30 times independently for statistical significance. The CPU time of the FAC algorithm was recorded on a 2.8 GHz Pentium IV PC.

4.1 Comparison with Existing Algorithms

The performance of the proposed FAC algorithm is compared with that of existing algorithms presented in [36], [38] and [43], which are the conventional Monte Carlo (EMC) algorithm, the genetic algorithm (GA) [36], the ant colony optimization (ACO) algorithm [38], and the immune algorithm (IA) [43]. The reason for choosing these algorithms is that their models and tests are the same as the ones used in this chapter. Table 2 compares the average performance of the IA, the ACO and the proposed FAC algorithm, in terms of the average time required (*AvgT*), the average energy evaluations (*A.E.E*), and the success rate (%ok. Table 3 compares the best time (*BestT*), the best energy evaluations (*B.E.E*), and the best number of iterations (*B.N.I*) among the FAC, the EMC, the GA, and the IA.

Table 2. Comparison of the average performance in solving the 2D-HP problems

No.	*l*	E^*	FAC	IA	ACO

			AvgT(sec.)	*A.E.E*	%ok	*A.E.E*	%ok	*AvgT*	%ok
1	18	-9	3.17703	115384	**100**	**18085.8**	**100**	--	--
2	18	-8	0.0967667	**3149**	**100**	69210	**100**	--	--
3	20	-10	0.264167	**8107**	**100**	41724.2	**100**	--	--
4	20	-9	**0.103667**	**2981**	**100**	23710	**100**	**< 1 sec.**	**100**
5	24	-9	1.2833	**32159**	**100**	69816.7	**100**	**< 1 sec.**	**100**
6	25	-8	3.90027	**93883**	**100**	269513.9	**100**	**< 1 sec.**	**100**
7	36	-14	**1.25527**	**18683**	**100**	2032504	**100**	4 sec.	**100**
8	48	-23	**28.922**	**331103**	**100**	6403985	56.67	1 min.	**100**

–The corresponding values are unavailable in the references [36], [38] or [43].

In Table 2, the mean values in the bold denote the best results of the three algorithms. Except for sequence 1, the average function evaluations of the FAC are much smaller than those of the IA. Moreover, the FAC has successfully found the best protein conformation in all the tests, while the IA has only managed to solve sequence 8 with a 56.67% success rate. Compared with the ACO, the average execution time of the FAC in obtaining the best protein for short protein sequences is not significantly longer, but it takes a shorter time for longer sequences such as Nos. 7 and 8.

In Table 3, among the best values of all algorithms, the FAC is seen much faster than the EMC and the GA in solving the sequences listed. Only are the best energy evaluations to sequence 6 slightly larger than that of the IA. It can be seen that the FAC algorithm developed in this chapter can solve the given protein folding problems in a very shortest period of time.

Table 3. Comparisons of the best performance in solving the 2D-HP problems

No.	*l*	E^*	FAC			EMC	GA	IA
			BestT(sec.)	*B.E.E*	*B.N.I*	*B.E.E*	*B.E.E*	*B.E.E*
4	20	-9	0.015	**169**	17	9374	30492	1925
5	24	-9	0.078	**1703**	171	6929	30491	2479
6	25	-8	0.234	5463	547	7202	20400	**4212**
7	36	-14	0.031	**234**	24	12447	301339	43416
8	48	-23	0.797	**9102**	92	165791	126547	37269

4.2 Analysis on Different Parameter Values

The influence of parameters in the FAC algorithm is also tested in order to assess the best group of values of the parameters, including the number of ants *m*, the heuristic reinforcement value β, and the global pheromone evaporation rate ρ. Fig. 8 shows the trends of different parameter values for sequences 1 to 7.

1) The heuristic reinforcement value β

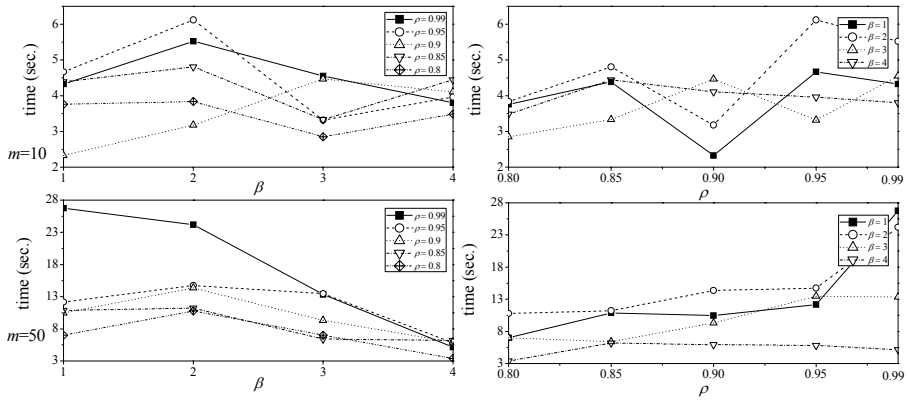
Fix the values of m and ρ. When β increases, the time needed to obtain solutions becomes shorter to sequences 1 to 5. Note that sequence 6 is distinctive in the sequences and it achieves the best result when $\beta = 1$.

2) The pheromone evaporation rate ρ

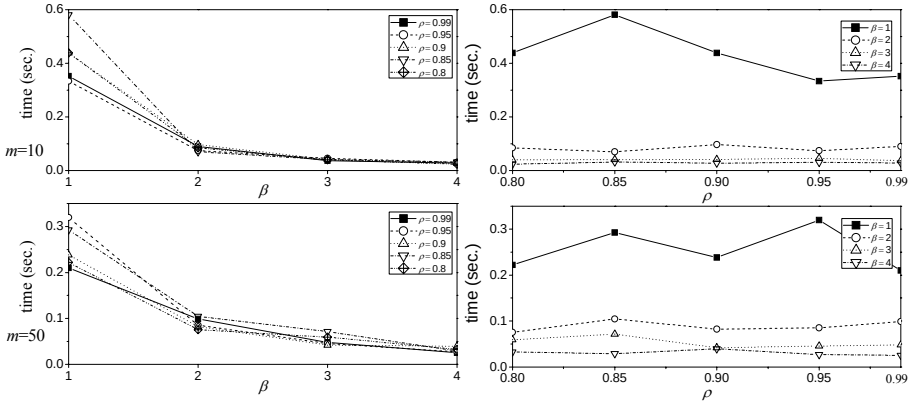
If the pheromone evaporation rate ρ is about 0.9, the performance of the FAC is high in most test cases. Overall, the influence of ρ is not so significant as β.

3) The number of ants *m*

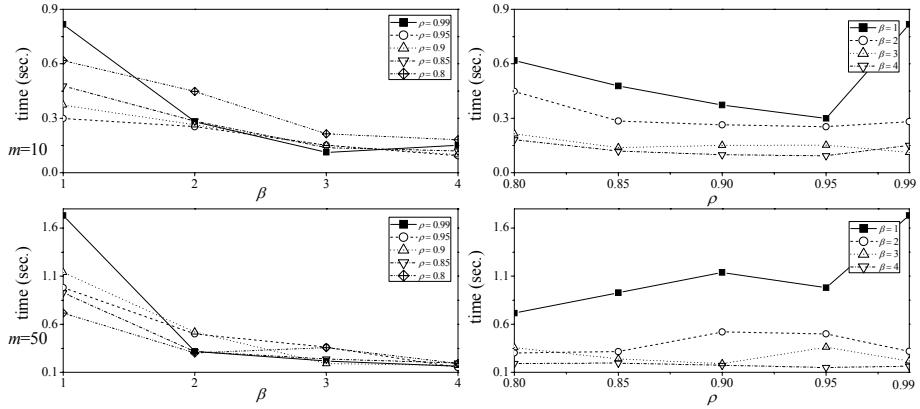
A large number of ants provide a higher insurance of finding the best conformation, but it slows down the algorithm. However, a small number of ants may induce early convergence to sub-optima. A proper number of ants is generally dependent upon the length of the protein sequence. For short protein sequences, m = 10 is enough. However, for long sequences such as the one with 48 amino acids, more ants (e.g., m= 100) are needed. Fig. 9 illustrates the convergent states (?) in the 30 independent tests of sequence 8. Conformations with 12 or 13 H-H bonds are always found in the first iteration. As the optimization continues, it takes more time to improve. The fastest search for the optimum folding of sequence 8 in the 30 tests was 92 iterations, while the worst one needed more than 10,000 iterations.



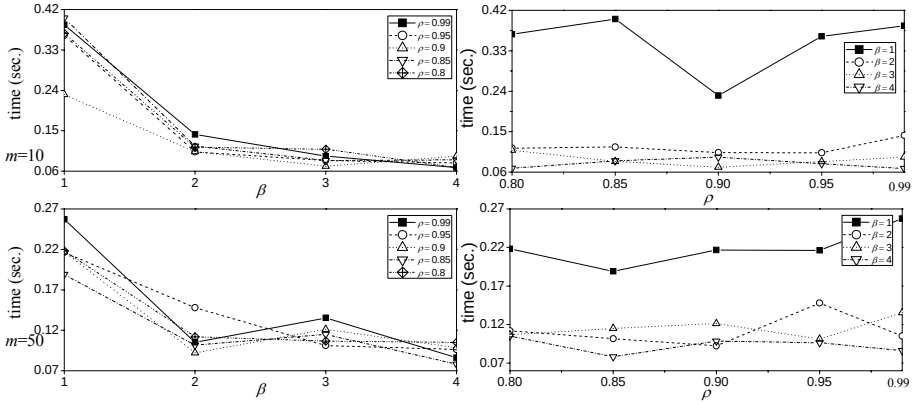
(a) Sequence 1 with m = 10 and m = 50



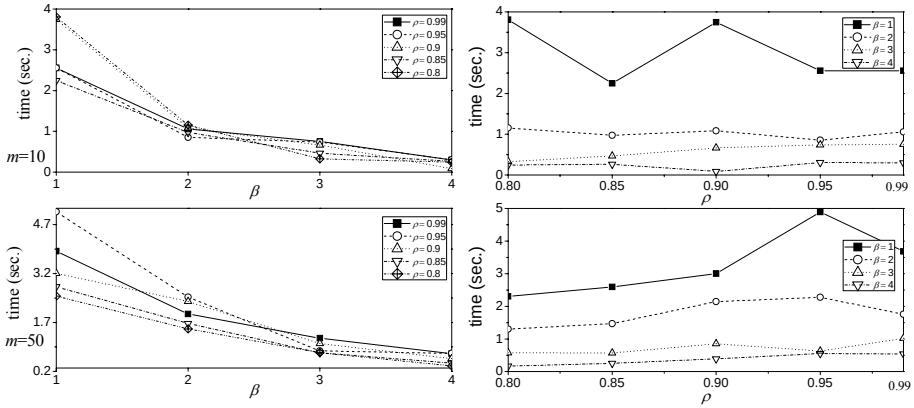
(b) Sequence 2 with m = 10 and m = 50



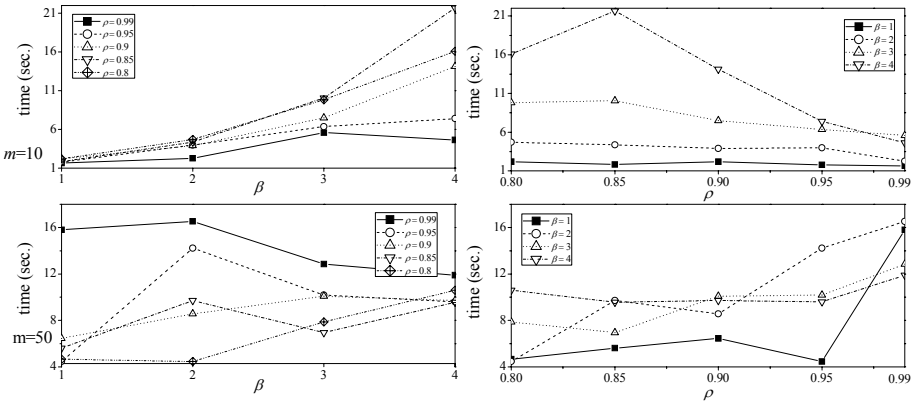
(c) Sequence 3 with m = 10 and m = 50



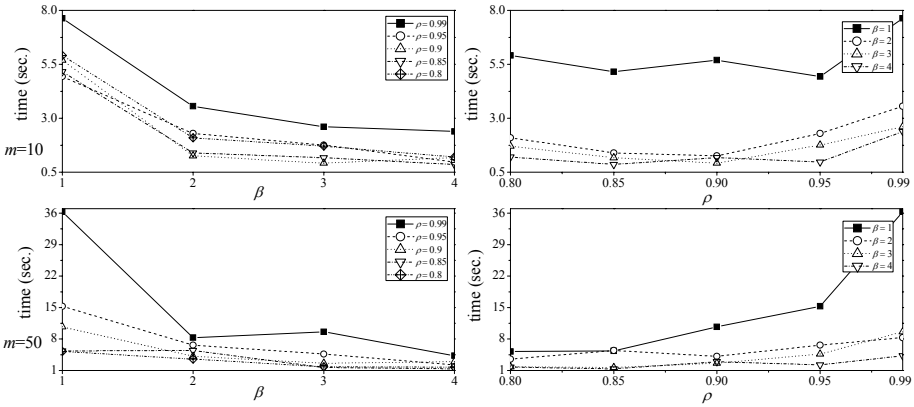
(d) Sequence 4 with m = 10 and m = 50



(e) Sequence 5 with m = 10 and m = 50



(f) Sequence 6 with m = 10 and m = 50



(g) Sequence 7 with m = 10 and m = 50

Fig. 8. Analysis of the FAC algorithm with various parameter values

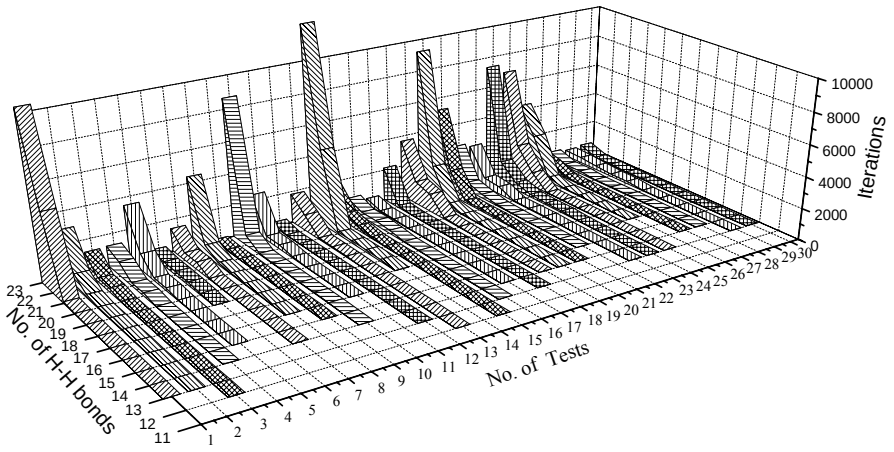


Fig. 9. Convergence in 30 independent tests to Sequence 8.

4.3 Analysis of Heuristic Information to Polar Amino Acids

In the proposed FAC algorithm, there is heuristic information for folding polar amino acids, which is different from that used in the ACO algorithm [38]. The performance of the FAC algorithm is compared with or without heuristic information to polar amino acids. The results are tabulated in Table 4. With the same parameter settings, the algorithm without heuristic information to polar amino acids is slower than the one with the heuristic information in all test cases. The results demonstrate that the heuristic information proposed in this chapter is effective.

Table 4. Comparisons on whether using heuristic information to polar amino acids

No.	*l*	E^*	FAC (use)			FAC (not use)		
			AvgT(sec.)	*A.E.E*	%ok	*AvgT*(sec.)	*A.E.E*	%ok
1	18	-9	**3.17703**	**115384**	100	7.69107	272580	100
2	18	-8	**0.0967667**	**3149**	100	0.181733	5903	100
3	20	-10	**0.264167**	**8107**	100	0.3943	11820	100
4	20	-9	**0.103667**	**2981**	100	0.116833	3271	100
5	24	-9	**1.2833**	**32159**	100	1.5146	36870	100
6	25	-8	**3.90027**	**93883**	100	4.14727	95673	100
7	36	-14	**1.25527**	**18683**	100	2.3422	33789	100
8	48	-23	**28.922**	**331103**	100	334.755	3756947	100

5 Conclusions

This chapter has presented a flexible ant colony algorithm for the protein folding problem. This FAC algorithm is based on the 2-dimensional square lattice hydrophobic-polar model, which is a highly abstract model for protein folding structures. Ants in the FAC algorithm start from the middle of the lattice and construct protein folding from the middle of the protein sequence. Pheromones are released to the directed arcs connecting adjacent squares in the lattice. Local pheromone update as well as global pheromone update mechanisms are also implemented. By using effective heuristic and pheromone method for selection, the proposed FAC algorithm can solve the PFP fast as shown by the test cases. Comparison with some well-known PFPalgorithms has highlighted superior performance of

the proposed FAC algorithm.

References

1. Chandru V, Dattasharma A, Kumar VSA (2003) The algorithms of folding proteins on lattices. Discrete Applied Mathematics 127:145–161

2. Mount DW (2001) Bioinformatics: sequence and genome analysis. Cold Spring Harbor Laboratory Press

3. Zaki MJ, Jin S, Bystroff C (2003) Mining residue contacts in proteins using local structure predictions. IEEE Trans on Systems Man and Cybernetics -Part B 33(5):789–810

4. Chen SS (2001) A localized protein-folding problem. International Journal of Intelligent Systems 16:449–457

5. RCSB (Research Collaboratory for Structural Bioinformatics). Protein Data Bank http://www.rcsb.org/pdb/

6. Anfinsen CB, Haber E, Sela M, White FH (1961) The kinetics of formation of native ribonuclease during oxidation of the reduced polypeptide chain. Proc Natl Acad Sci USA 47:1309–1314

7. Anfinsen CB (1973) Principles that govern the folding of proteins chains. Science 181:223–230

Flexible Protein Folding by Ant Colony Optimization 19

8. Woldawer A, Miller M, Jaskolski M, Sathyanarayana BK, Baldwin E, Weber IT, Selk LM, Clawson L, Schneider J, Kent SB (1989) Conserved folding in retroviral proteases: crystal structure of a synthetic HIV-1 protease. Science 245:616–621

9. Clore GM, Gronenborn AM(1991) Comparison of the solution nuclear magnetic resonance and X-ray crystal structures of human recombinant interleukin-1 β . J Mol Biol 221:47–53

10. Unger R, Moult J (1993) Finding the lowest free energy conformation of a protein is an NP-hard problem: proof and implications. Bulletin of Mathematical Biology 55:1183–1198

11. Govindarajan S, Goldstein RA (1998) On the thermodynamic hypothesis of protein folding. Proc Natl Acad Sci USA 95:5545–5549

12. Ngo JT, Marks J (1992) Computational complexity of a problem in molecularstructure prediction. Protein Engineering 5(4):313–321

13. Fraenkel AS (1993) Complexity of protein folding. Bulletin of Mathematical Biology 55(6):1199–1210

14. Guo YZ, Feng EM (2006) Exploration of two-dimensional hydrophobic-polar lattice model by combining local search with elastic net algorithm. Journal of Cbemical Physics 125: 154102

15. Lesh N, Mitzenmacher M, Whitesides S (2003) A complete and effective move set for simplified protein folding. RECOMB'03 Berlin Germany 188–195

16. Dill KA, Bromberg S, Yue K, Fiebig KM, Yee DP, Thomas PD, Chan HS (1995) Principles of protein folding - a perspective from simple exact models. Protein Science 4:561–602

17. Hart WE, Istrail S (1995) Fast protein folding in the hydrophobic-hydrophilic model within three-eights of optimal. STOC'95 USA 157–168

18. Chan HS, Dill KA (1994) Transition states and folding dynamics of proteins and heteropolymers. J Chem Phys 100(12):9238–9257

19. Hoque MT, Chetty M, Dooley LS (2006) A hybrid genetic algorithm for 2D FCC hydrophobic-hydrophilic lattice model to predict protein foldings. AI 2006 LNAI 4304: 867–876

20. Agarwala R, Batzoglon S, Danˇc′ık V, Decatur SE, Farach M, Hannenhalli S, Skiena S (1997) Local rules for protein folding on a triangular lattice and generalized hydrophobicity in the HP model. Proceedings of the eighth annual ACM-SIAM symposium on discrete algorithms USA 390–399

21. Decatur SE (1996) Protein folding in the generalized hydrophobic-polar model on the triangular lattice. MIT LCS Technical memo: MIT-LCS-TM-559

22. Liu J, Wang LH, He LL, Shi F (2005) Analysis of toy model for protein folding based on particle swarm

optimization algorithm. ICNC 2005 LNCS 3612:636–645
23. Stillinger FH, Head-Gordon T, Hirshfeld CL (1993) Toy model for protein folding. Phys Rev E 48(2):1469–1477
24. Liang F, Wong WH (2001) Evolutionary Monte Carlo for protein folding simulations. J Chem Phys 115(7):3374–3380
25. Backofen R, Will S, Clote P (2000) Algorithmic approach to quantifying the hydrophobic force contribution in protein folding. PSB'2000 92–103
26. Bastolla U, Frauenkron H, Gerstner E, Grassberger P, Nadler W (1998) Testing a new Monte Carlo algorithm for protein folding. Proteins: Structure Function and Genetics 32:52–66
27. Hsu HP, Mehra V, W. Nadler, Grassberger P (2003) Growth algorithm for lattice heteropolymers at low temperatures. J Chem Phys 118:444
28. Ramakrishnan R, Ramachandran B, Pekny JF (1997) A dynamic Monte Carlo algorithm for exploration of dense conformational spaces in heteropolymers. J Chem Phys 106(6):2418–2425
29. Tantar AA, Melab N, Talbi EG, Parent B,Horvath D (2007) A parallel hybrid genetic algorithm for protein structure prediction on the computational grid. Future Generation Computer Systems 23:398–409
30. Arunachalam J, Lanagasabai V, Gautham N (2006) Protein structure prediction using mutually orthogonal Latin squares and a genetic algorithm. Biochemical and Biophysical Research Communications 342:424–433
31. Hoque MT, Chetty M, Dooley LS (2005) A new guided genetic algorithm for 2D hydrophobic-hydrophilic model to predict protein folding. CEC 2005 1:259–266
32. Bui TN, Sundarraj G (2005) An efficient genetic algorithm for predicting protein tertiary structures in the 2D HP Model. GECCO'05 USA 385–392
33. Cust′odio FL, Barbosa HJC, Dardenne LE (2004) Investigation of the threedimensional lattice HP protein folding model using a genetic algorithm. Genetics and Molecular Biology 27(4):611–615
34. Krasnogor N, Hart WE, Smith J, Pelta DA (1999) Protein structure prediction with evolutionary algorithms. Proceedings of the Genetic and Evolutionary Computation Conference 1596–1601
35. Pedersen JT, Moult J (1996) Genetic algorithms for protein structure prediction. Current Opinion in Structural Biology 6(2):227-231
36. Unger R, Moult J (1993) Genetic algorithms for protein folding simulations. J Mol Biol 231(1):75–81
37. Daeyaert F, Jonge MD, Koymans L, Vinkers M (2007) An ant algorithm for the conformational analysis of flexible molecules. Journal of Computational Chemistry 28(5):890–898
38. Shmygelska A, Hoos HH (2005) An ant colony optimisation algorithm for the 2D and 3D hydrophobic polar protein folding problem. BMC Bioinformatics 6:30
39. Chu D, Till M, Zomaya A (2005) Parallel ant colony optimization for 3D protein structure prediction using the HP lattice model. IPDPS'05 1–7
40. Shmygelska A, Hoos HH (2003) An improved ant colony optimisation algorithm for the 2D HP protein folding problem. Advances in Artificial Intelligence, Berlin Springer-Verlag LNAI 2671:400-417
41. Shmygelska A, Aguirre-Hernandez R, Hoos HH (2002) An ant colony optimization algorithm for the 2D HP protein folding problem. ANTS 2002 Berlin Springer-Verlag LNCS 2463:40–52
42. Cutello V, Nicosia G, Pavone M, Timmis J (2007) An immune algorithm for protein structure prediction on lattice models. IEEE Trans. on Evolutionary Computation 11(1):101–117
43. Cutello V, Nicosia G, Pavone M (2004) An immune algorithm with hypermacromutations for the Dill's 2D hydrophobic-hydrophilic model. CEC 2004 1: 1074–1080
44. Crescenzi P, Goldman D, Papadimitriou C, Piccolboni A, Yannakakis M (1998) On the complexity of protein folding. J Comp Biol 5(3):423–466

45. Dorigo M, St¨utzle T (2004) Ant Colony Optimization. Cambridge MA: MIT Press
46. Dorigo M, Gambardella LM (1997) Ant Colony System: A Cooperative Learning Approach to the Traveling Salesman Problem. IEEE Trans on Evol Comput 1(1):53–66
47. Dorigo M, Maniezzo V, Colorni A (1996) Ant system: optimization by a colony of cooperating agents. IEEE Trans on Systems Man and Cybernetics–Part B 26(1):29–41
48. Gambardella LM, Taillard ′ED, Agazzi G (1999) MACS-VRPTW: A Multiple Ant Colony System for Vehicle Routing Problems with Time Windows. In D. Corne, M. Dorigo, and F. Glover (Eds.). New Ideas in Optimization London McGraw Hill 63–76
49. Zhang J., Hu XM, Tan X, Zhong JH, Huang Q (2006) Implementation of an Ant Colony Optimization Technique for Job Shop Scheduling Problem. Transactions of the Institute of Measurement and Control 28(1):1–16
50. Zecchin AC, Simpson AR, Maier HR, Nixon JB (2005) Parametric Study for an Ant Algorithm Applied to Water Distribution System Optimization. IEEE Trans Evol Comput 9:175–191.